



Advances in Parallelizing Algebraic Multigrid

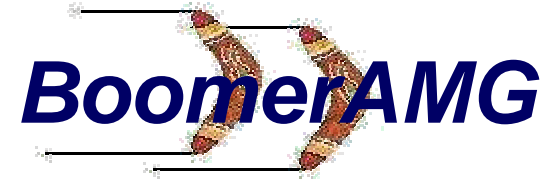
Van Emden Henson

*Center for Applied Scientific Computing
Lawrence Livermore National Laboratory*

Livermore, California, USA

July 5, 1999





The *BoomerAMG* Team

- Algorithm development, design, codewriting, maintenance
 - Van Emden Henson
 - Ulrike Meier Yang
- Algorithm development
 - Robert D. Falgout
 - Jim E. Jones
 - Andrew Cleary
- Consulting
 - John Ruge

Outline

- Parallelization of AMG
- Coarsening techniques
- Relaxation techniques
- Numerical results
- Conclusions



AMG has two phases:

- **Setup Phase**

- Select Coarse “grids,” $\Omega^{m+1}, m = 1, 2, \dots$

- Define **interpolation**, $I_{m+1}^m, m = 1, 2, \dots$

- Define **restriction** and **coarse-grid operators**

$$I_m^{m+1} = (I_{m+1}^m)^T \quad A^{m+1} = I_m^{m+1} A^m I_{m+1}^m$$

- **Solve Phase**

- Standard multigrid operations, e.g., V-cycle, W-cycle, FMG, etc

We must parallelize these steps:



- In The Setup Phase
 - Coarse Grid Selection
 - Construction of Prolongation operator, P
 - Construction of coarse-grid operators by Galerkin method, RAP , $R=P'$
- In The Solve Phase
 - Residual Calculation
 - Relaxation
 - Prolongation
 - Restriction

Parallelizing the Solve Phase

- In The Solve Phase
 - **Residual Calculation**
 - entails $Axpy$ Matvec: $y \leftarrow -aAx + by$.
 - **Relaxation**
 - Jacobi is essentially a Matvec
 - Gauß-Seidel is sequential, but hybrid (or chaotic) schemes may be employed
 - **Prolongation**
 - requires a Matvec (on a rectangular matrix)
 - **Restriction**
 - requires a MatvecT

Basic concept: Smooth error means “small” residuals



- Error that is slow to converge obeys:

$$e^{k+1} = (I - Q^{-1}A) e^k \quad ; \text{ hence } (I - Q^{-1}A) e \approx e \\ \Rightarrow Q^{-1}A e \approx 0 \Rightarrow r \approx 0$$

- Define: *i depends on j* (and *j influences i*) if

$$-a_{ij} \geq \theta \max_{k \neq i} \{-a_{ik}\}, \quad 0 < \theta \leq 1$$

- The set of dependencies of *i* is given by

$$S_i = \left\{ j : -a_{ij} > \theta \max_{j \neq i} -a_{ij} \right\}$$

- Smooth error varies slowly in the direction of dependence

Coarsening techniques

- classical Ruge- Stüben(RS) algorithm
- **Cleary-Luby-Jones-Plassman (CLJP) algorithm**
- **parallel Ruge-Stüben coarsening techniques**
- **Falgout-CLJP coarsening**

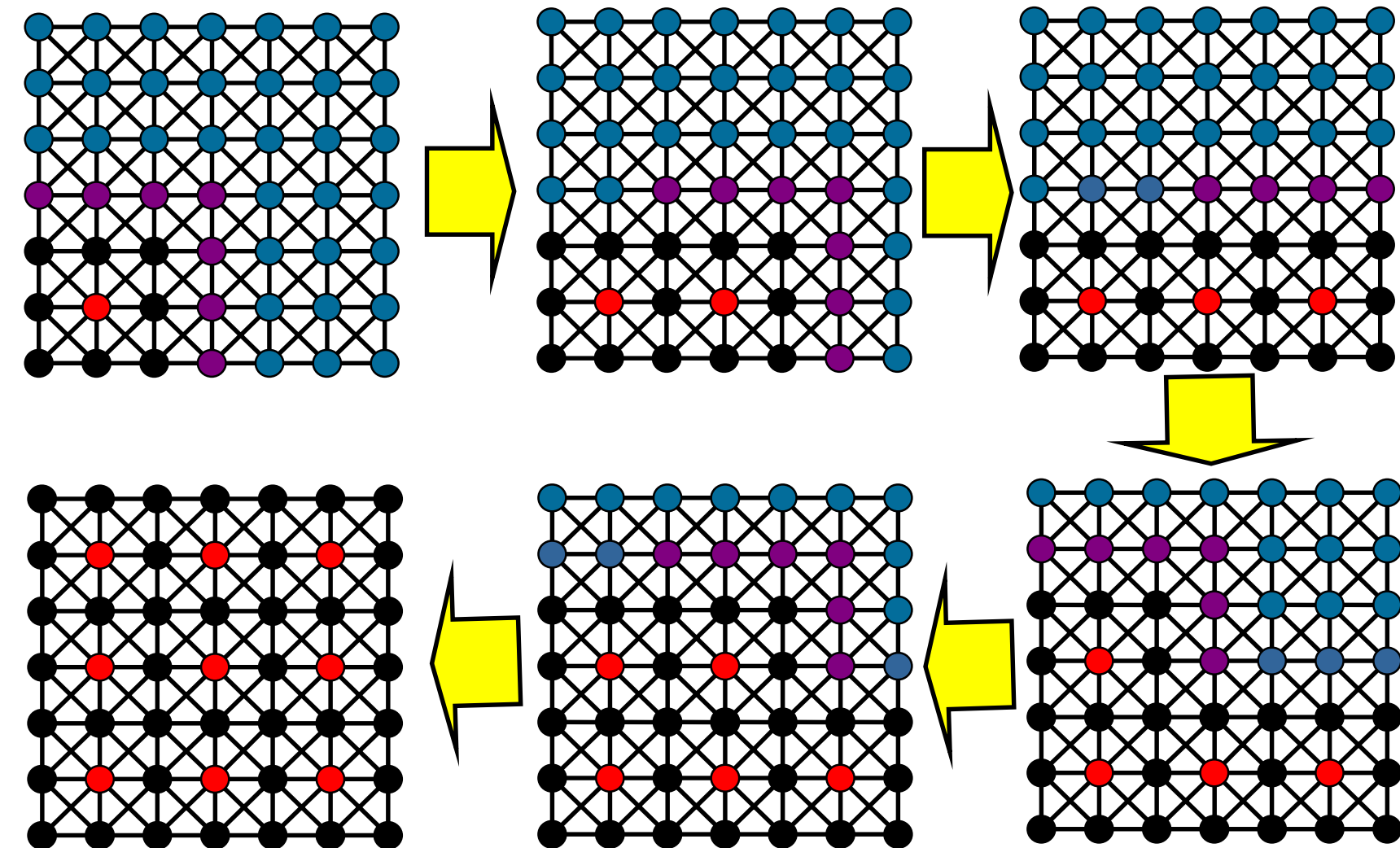
Choosing the Coarse Grid

- **Two Criteria**

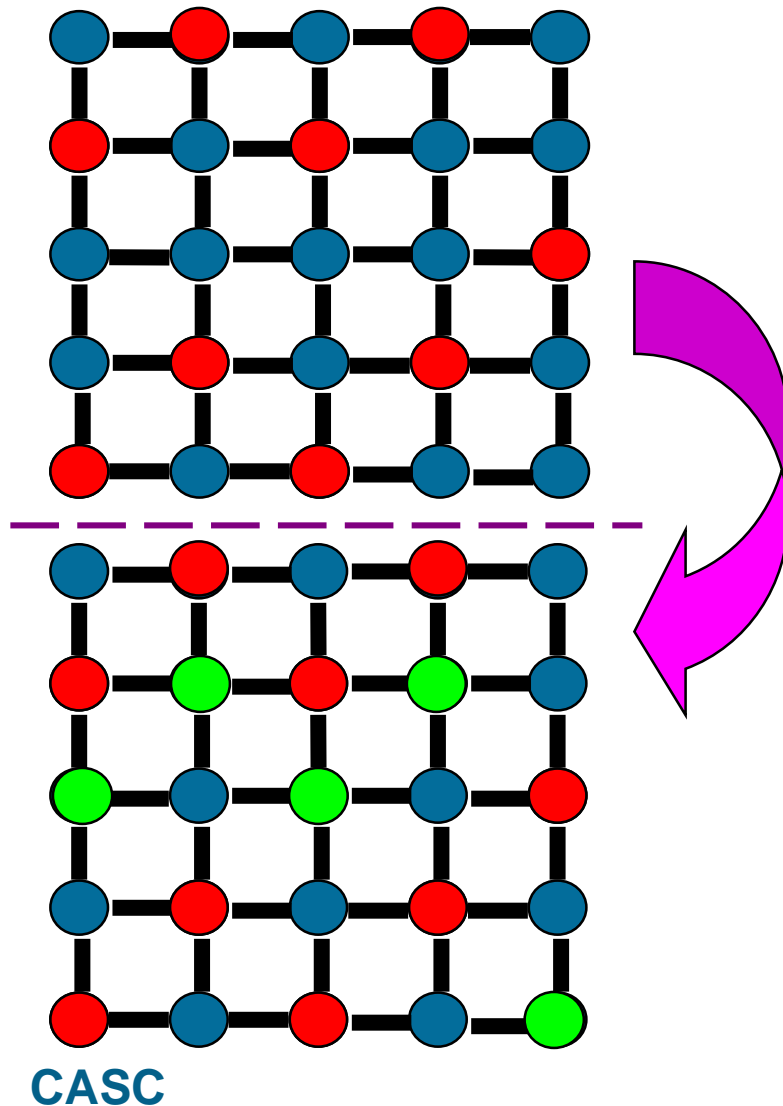
- **(C1)** For each $i \in F$, each point $j \in S_i$ should either be in C or should be strongly connected to at least one point in C_i
- **(C2)** C should be a maximal subset with the property that no two C -points are strongly connected to each other.

- Satisfying both (C1) and (C2) is sometimes impossible. We use (C2) as a guide while enforcing (C1).

The AMG coarse-grid selection algorithm is inherently sequential

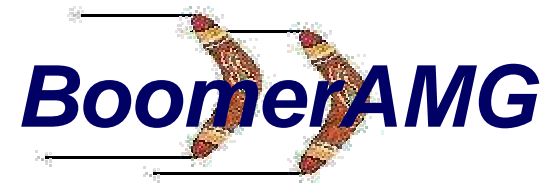


A second pass is needed to enforce (C1)



- First-pass coarsening of 5 point Laplacian , **periodic boundary conditions**
- Numerous ***F-F** dependencies among points not sharing common **C**-point*
- A second “coloring” pass is made, changing ***F***-points to **C**-points, as needed, to ensure (C1).

Parallel Ruge-Stüben Coarsening

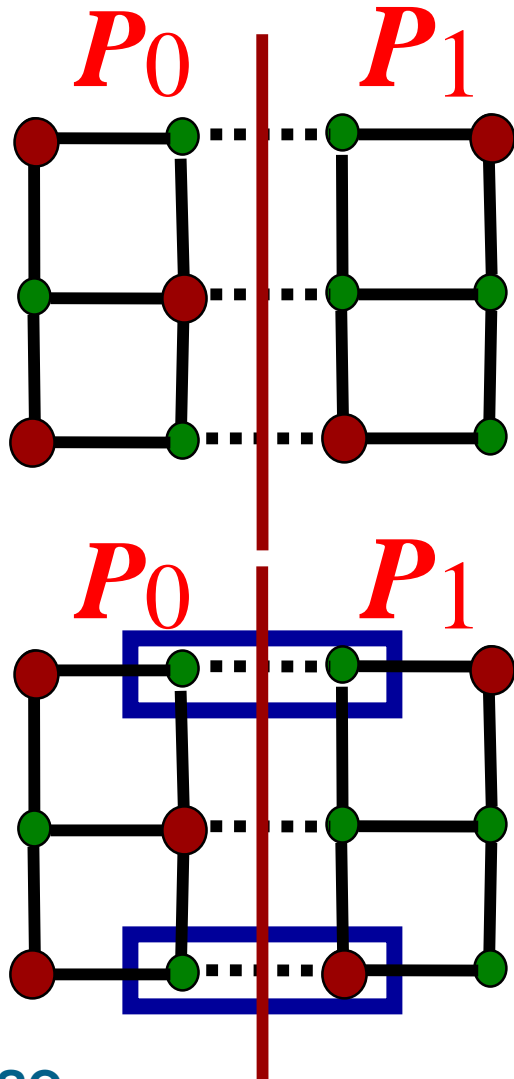
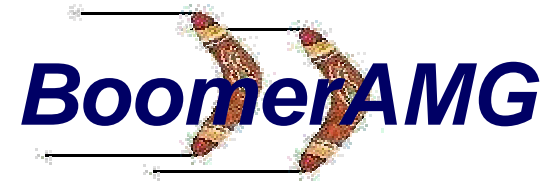


- One approach to coarsening in parallel: perform the standard Ruge- Stüben algorithm on each processor.
- Various treatments possible at processor boundaries.
- Yields **processor dependent coarsenings**, and will not produce the same results for different numbers of processors.

Measures

- **Measure = number of strong influences.**
- **Possible treatments of the measure for parallel Ruge-Stüben coarsening:**
 - **Determine measures locally, no communication between processors (RS)**
 - **Use the ‘correct’ measures, i.e., take into account off-processor connections (RScm)**

Parallel RS coarsening: boundary treatment: RS

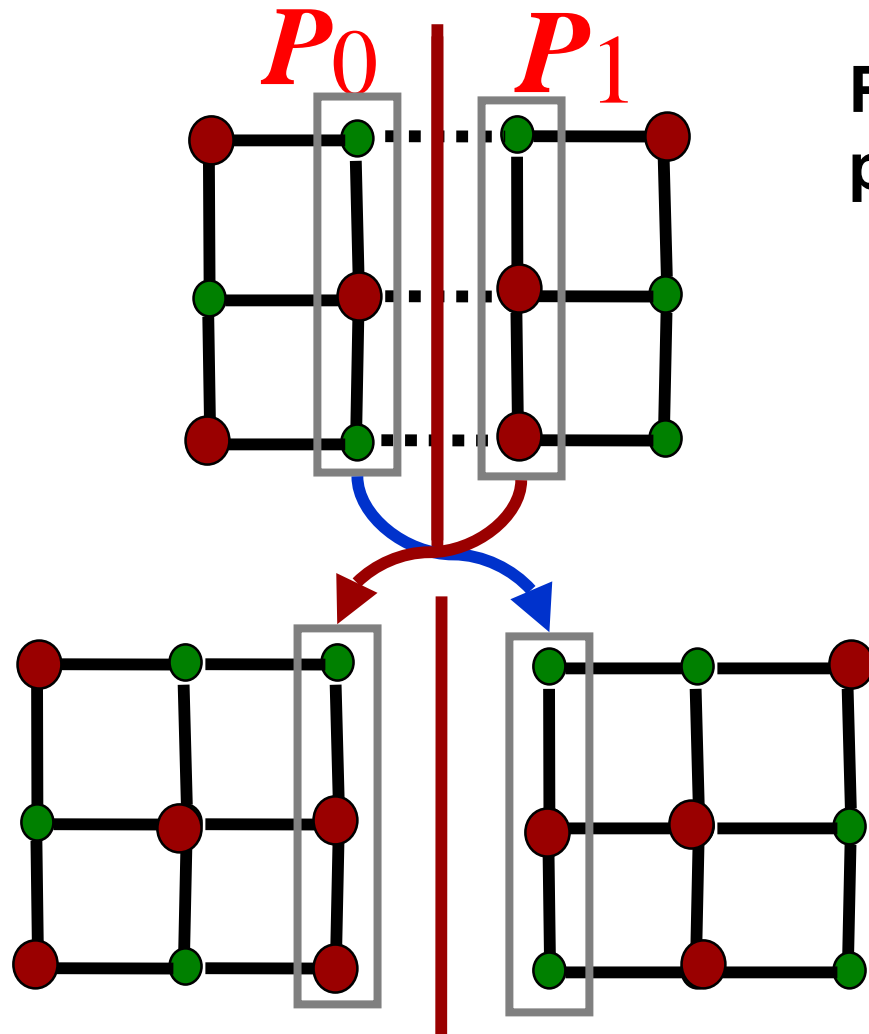


Perform first and second passes on each processor

Method 1: Do nothing.
Accept the coarsening provided by the independent processors.

Problem: Leaves $F \Leftrightarrow F$ dependencies without mutual C -points

Parallel RStüben coarsening: boundary treatment (**RS2b**)

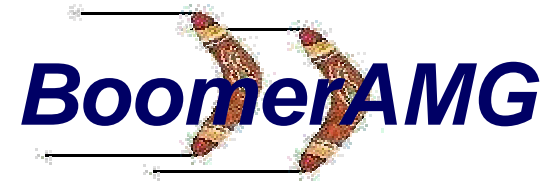


Perform first pass on each processor

Perform second pass locally on each processor, augmented by boundary points from neighbor

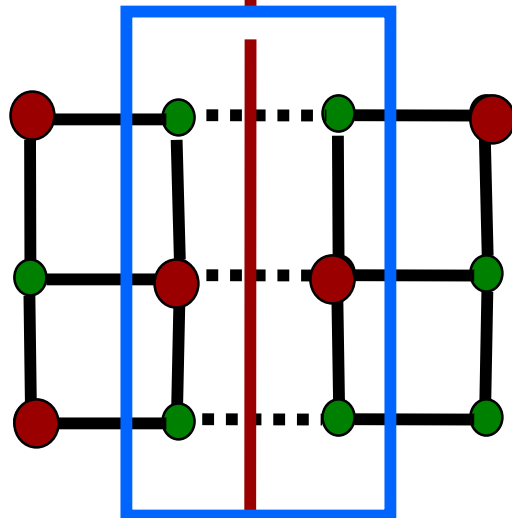
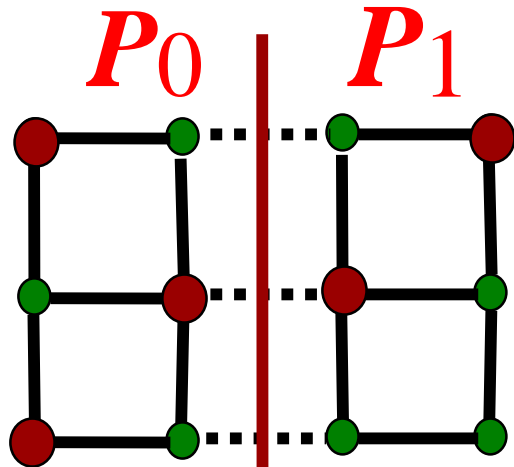
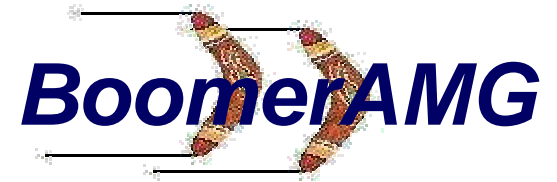
Choices must be made about how to resolve conflicting decisions among processors

Boundary conflict resolution



- Methods to resolve conflicting coarsenings at processor boundaries:
 - Largest processor ID wins (RS3, RS2b)
may violate (C1)
 - keep all coarse points (RS3c)
does not violate (C1)
may yield “too many” coarse-points
giving high operator complexity

Parallel RS coarsening: boundary treatment (**RS3**)



Perform first and second pass on each processor

Perform a third pass, (a second “second pass”),
only on those points
adjacent to processor
boundaries

Choices must be made about
how to resolve conflicting
decisions among processors

Parallel Ruge-Stüben coarsening results



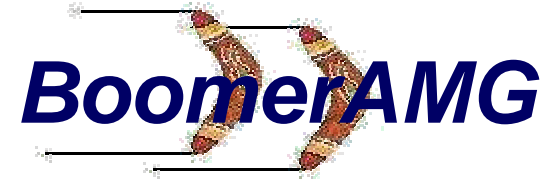
7 pt 3D Laplacian		
Procs.	Setup	Op. Cplx
1	20	4.91
2	30	5.25
4	48	5.71
8	79	6.23
16	119	6.75
32	194	6.98
64	360	7.34
128		
Procs	Solve	C.F.
1	36	0.065
2	40	0.081
4	43	0.111
8	48	0.210
16	389	0.246
32	3433	0.605
64	3352	0.384
128		

Ruge-Stüben coarsening is much faster and yields much better complexities than Cleary-LJP on the 7-pt Laplacian

Note that the solve times jump by orders of magnitude as problem grows. Parallel Ruge leads to large “coarsest” grids with direct solve.

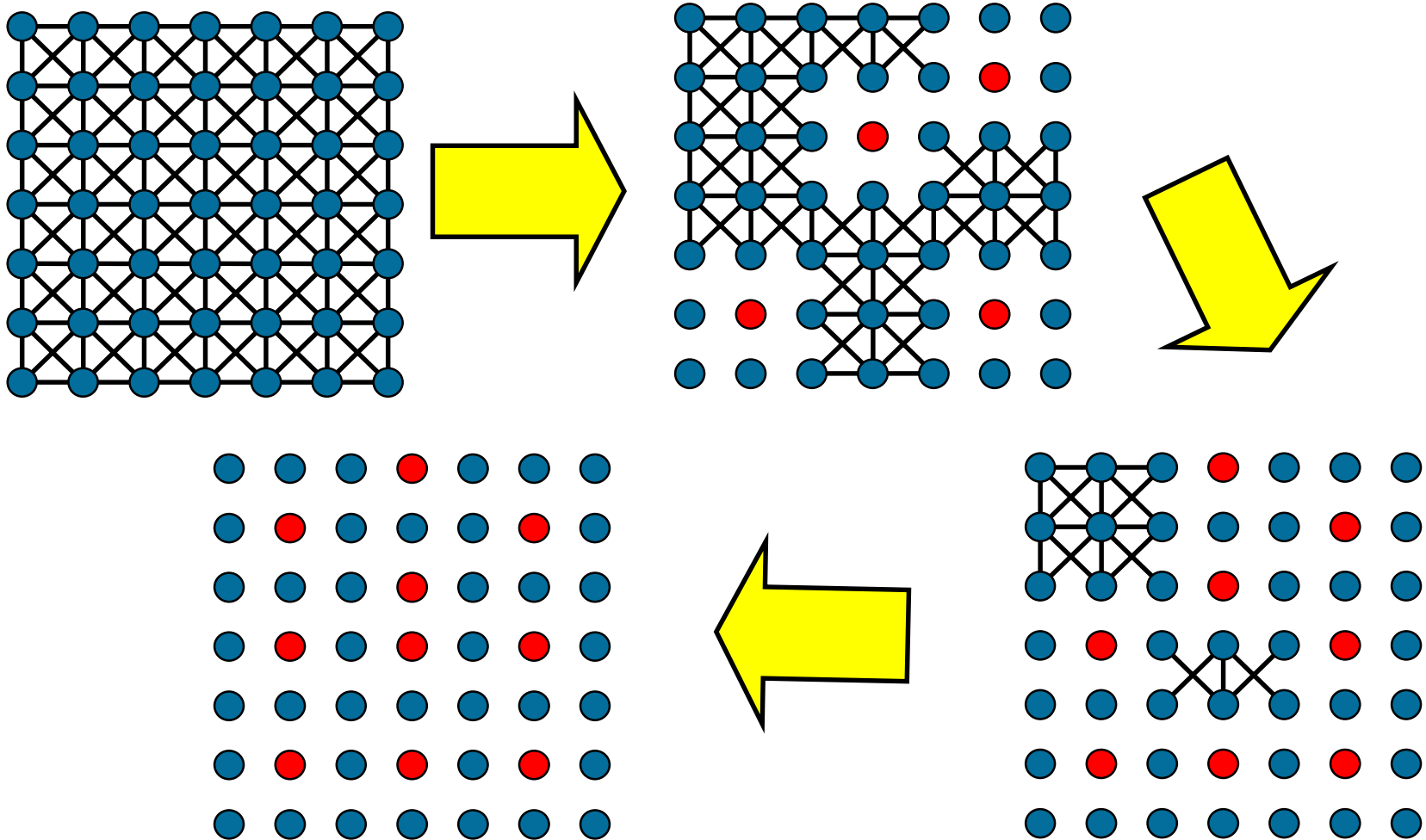
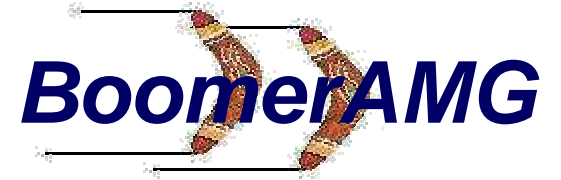
Solution: hybrid coarsening?

A new approach: the Cleary-LJP algorithm

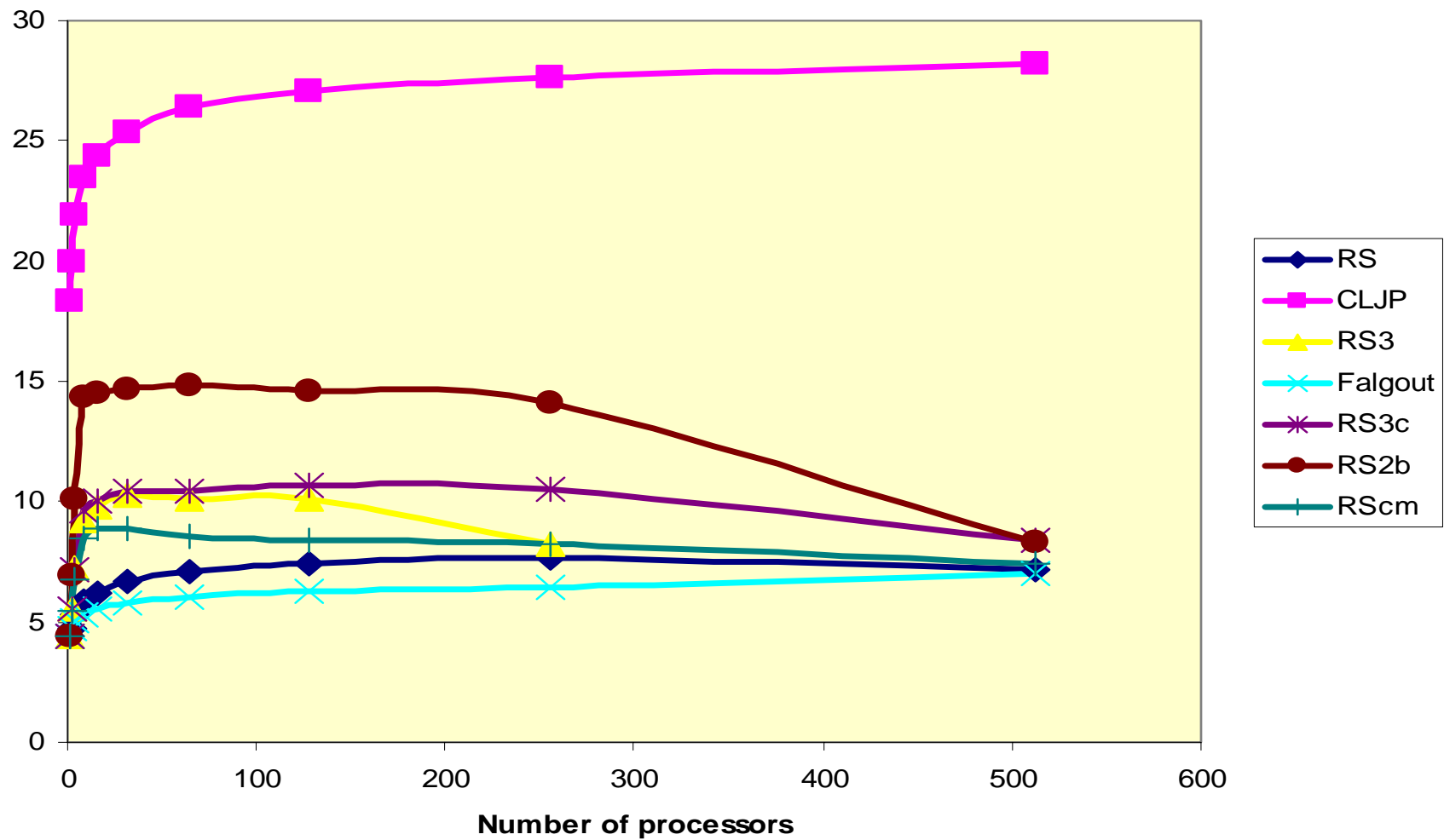


- The Ruge-Stüben algorithm is **inherently sequential**.
- A new algorithm was proposed by Andrew Cleary , following parallel-independent-set algorithms developed by Luby and later by Jones & Plassman
- Resulting coarsening algorithm (Cleary-LJP) is fully parallel, independent of the number of processors or processor topology. Serial prototype early 98, parallel code late 98.

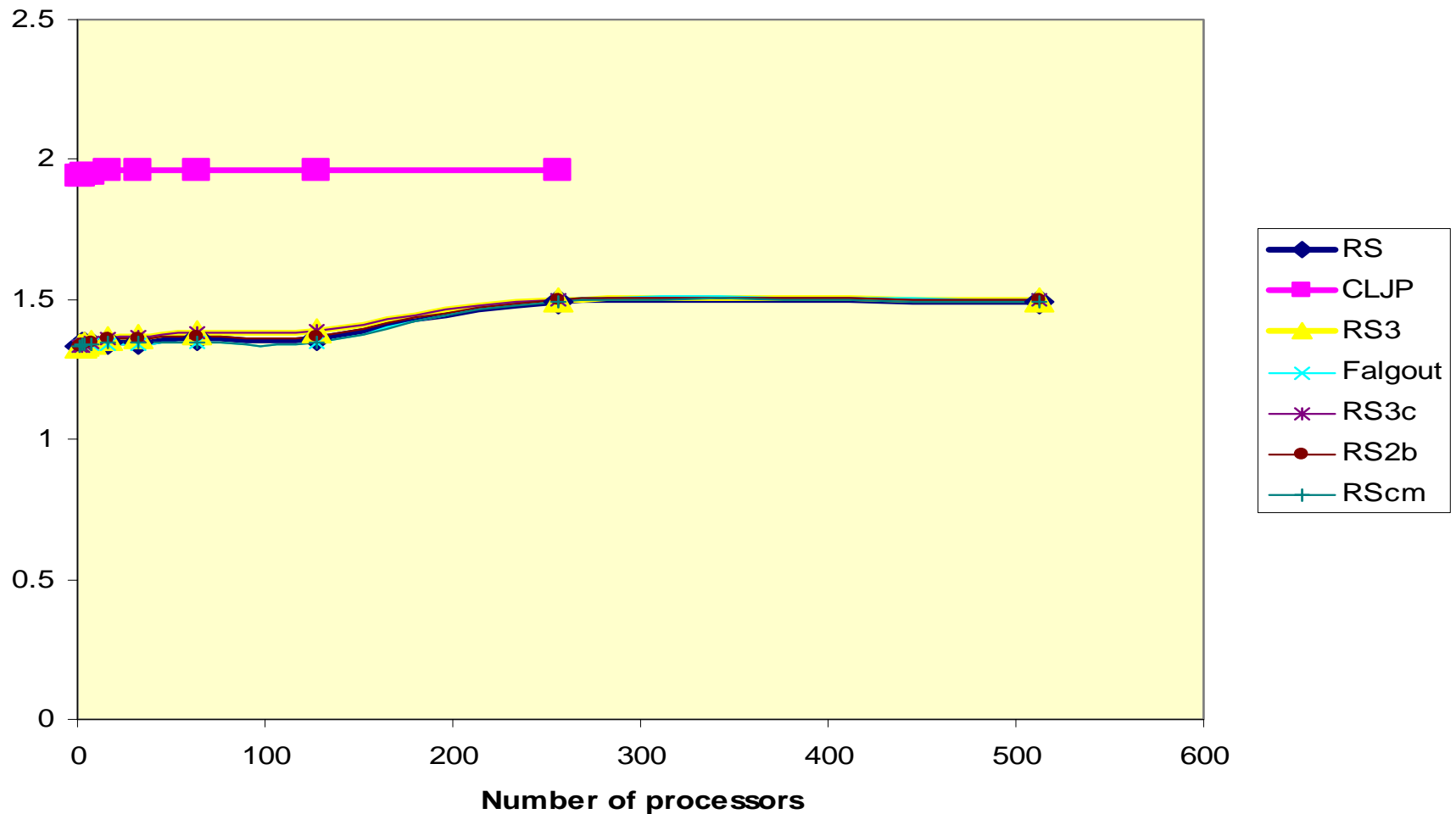
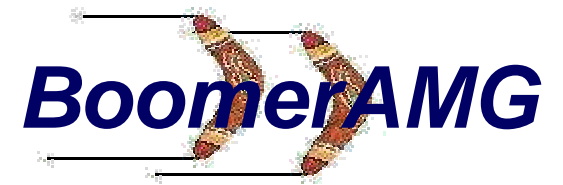
The C-LJP coarsening is fully parallel; independent of P



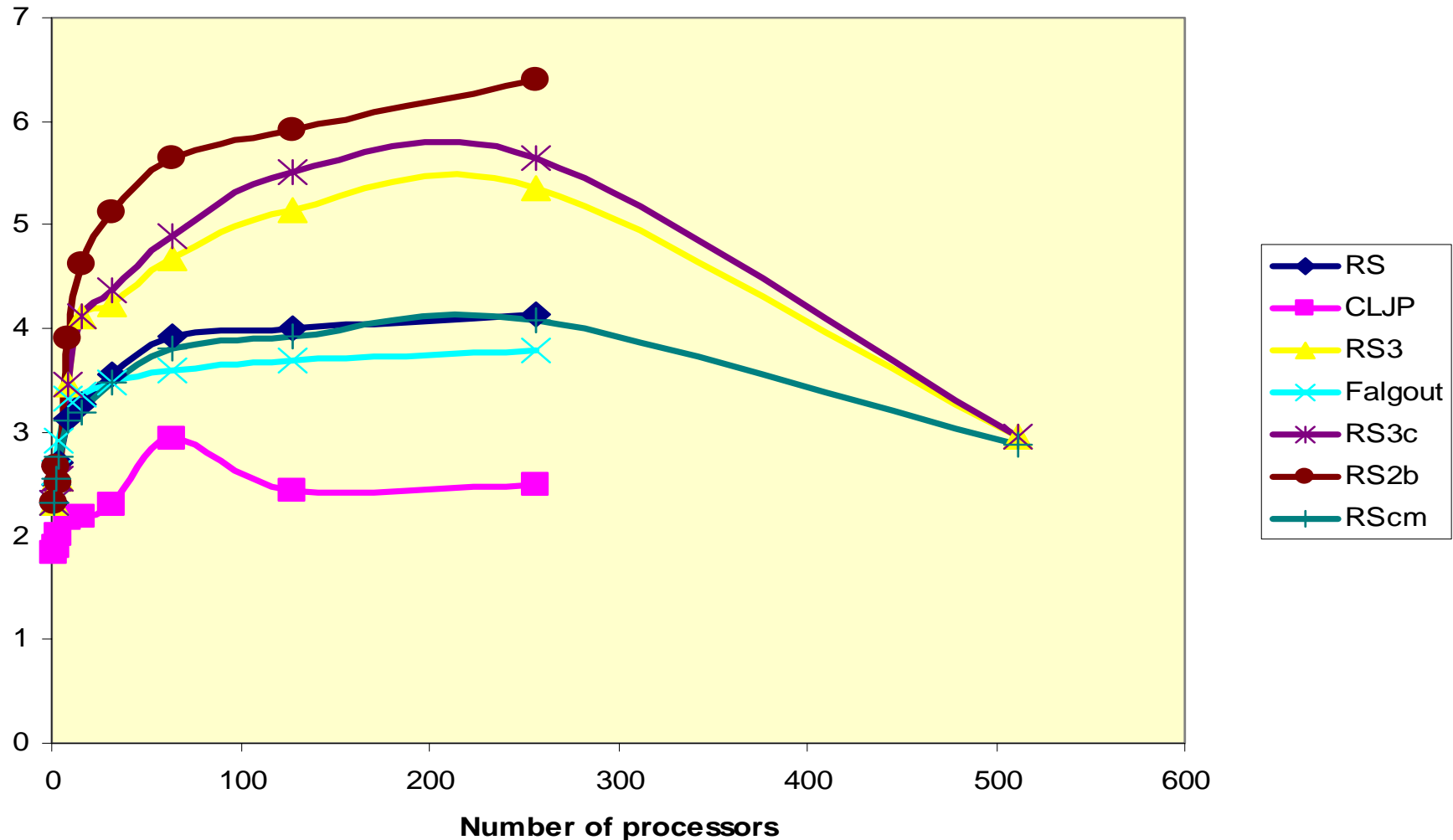
7 point 3D Laplacian: Operator Complexities



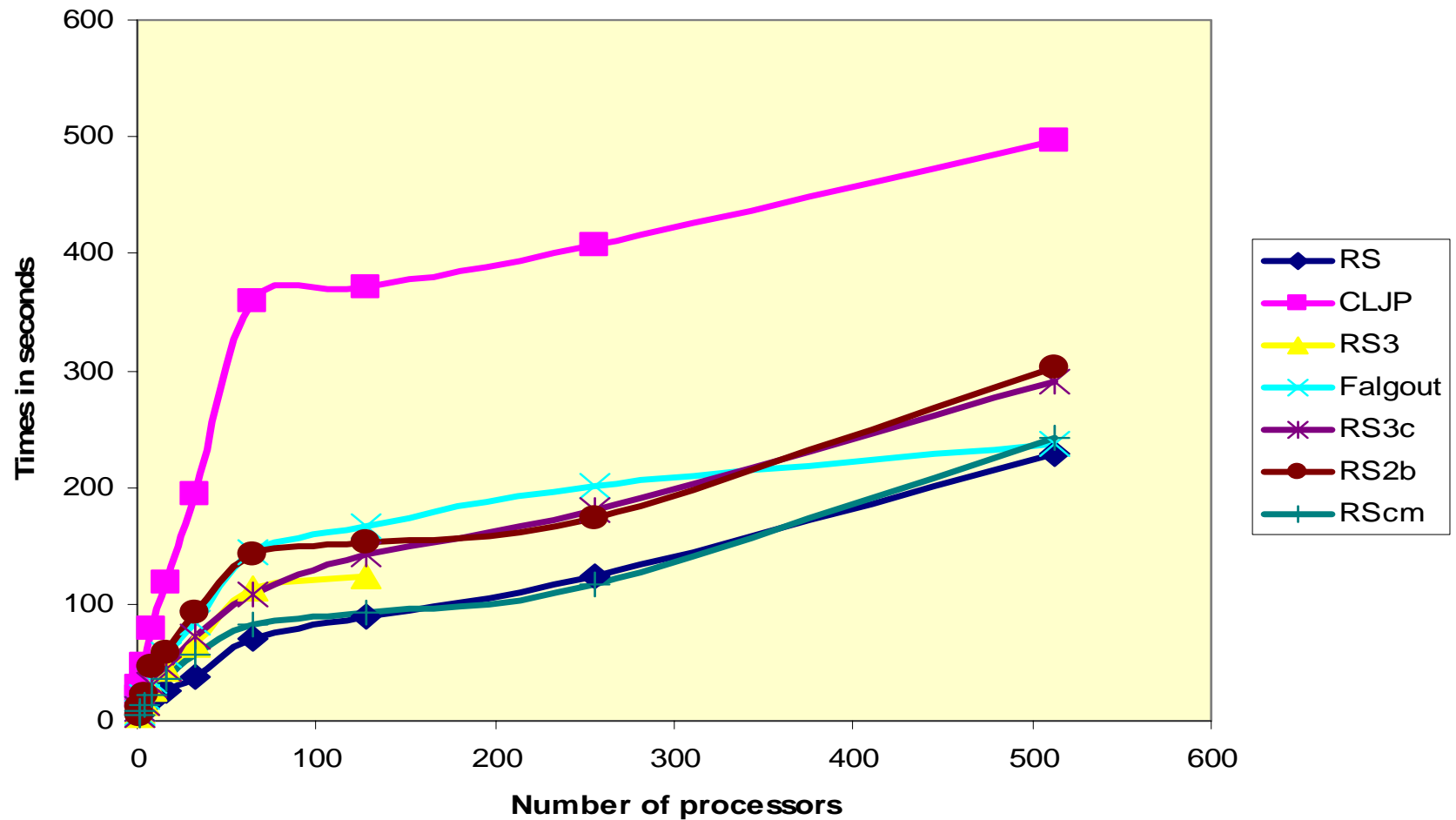
9 point 2D Laplacian: Operator complexities



27 point 3D Laplacian: Operator complexities

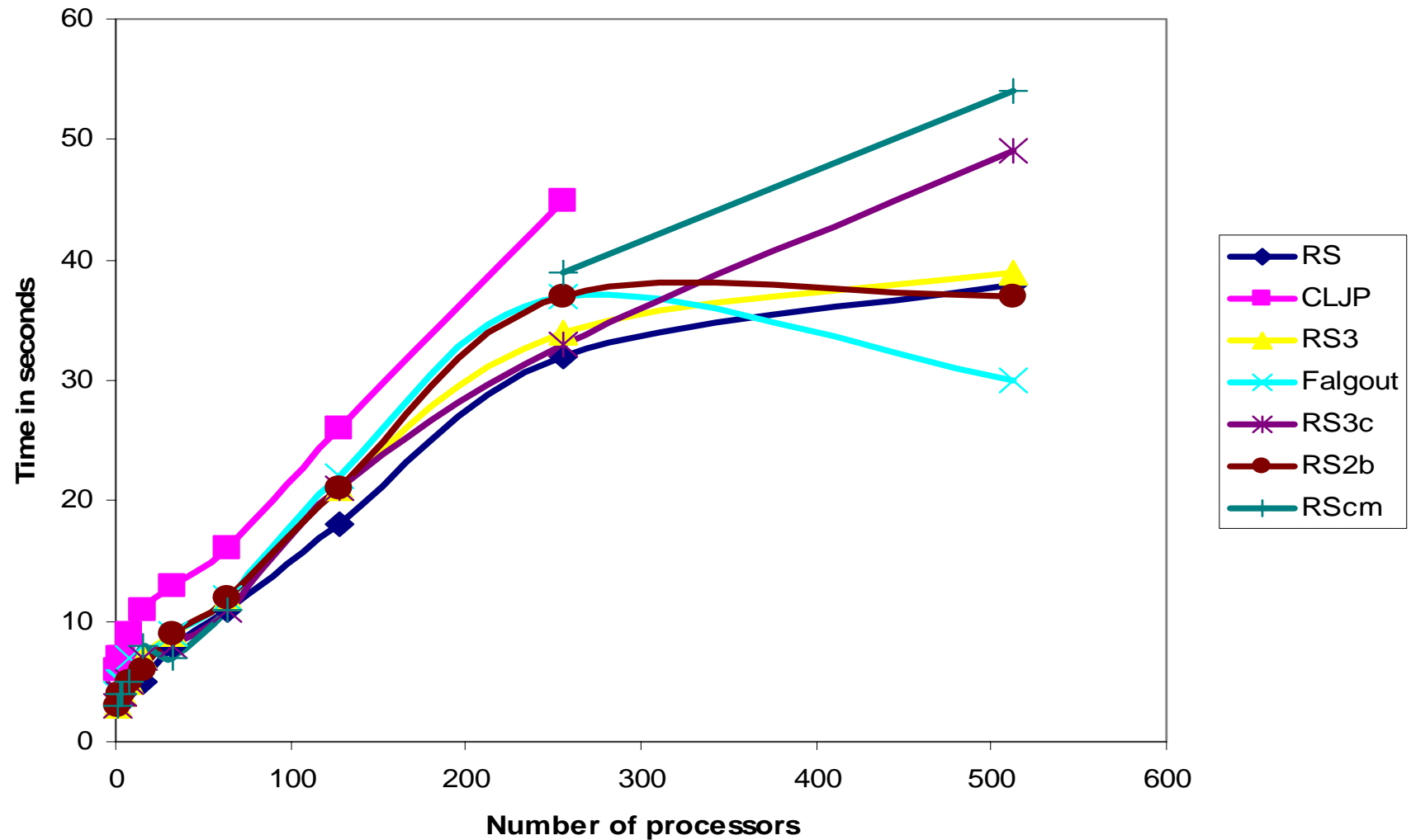


7 point 3D Laplacian: Setup times

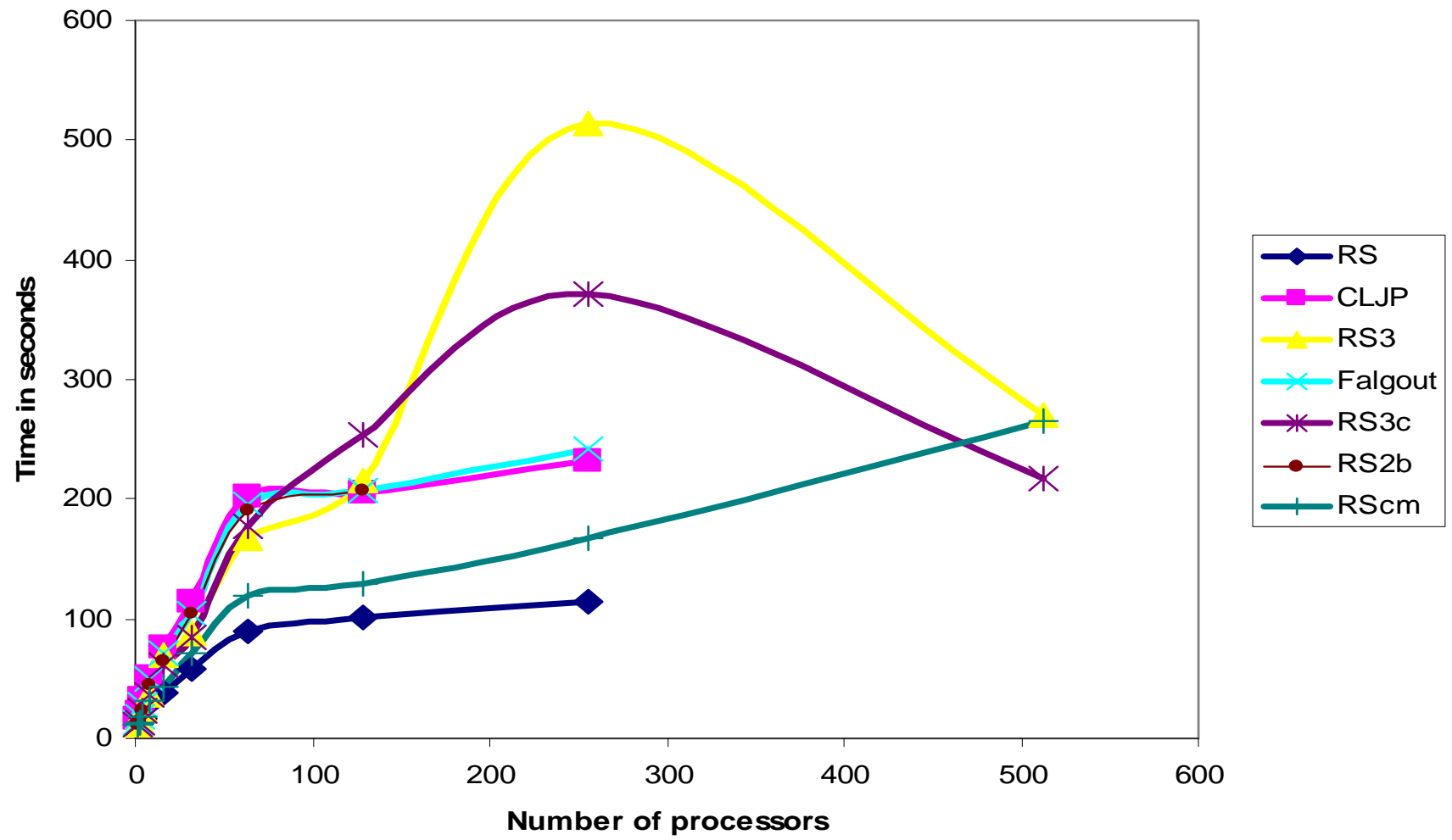
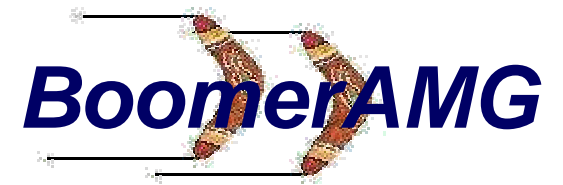


9 point 2D Laplacian: Setup times

BoomerAMG



27 point 3D Laplacian: Setup times



Relaxation techniques

- **Jacobi or weighted Jacobi**

$$x^{(n+1)} = (2-w)x^{(n)} + wD^{-1}(b - Ax^{(n)})$$

e.g. $w = \frac{1}{\|D^{-1/2}AD^{-1/2}\|}$

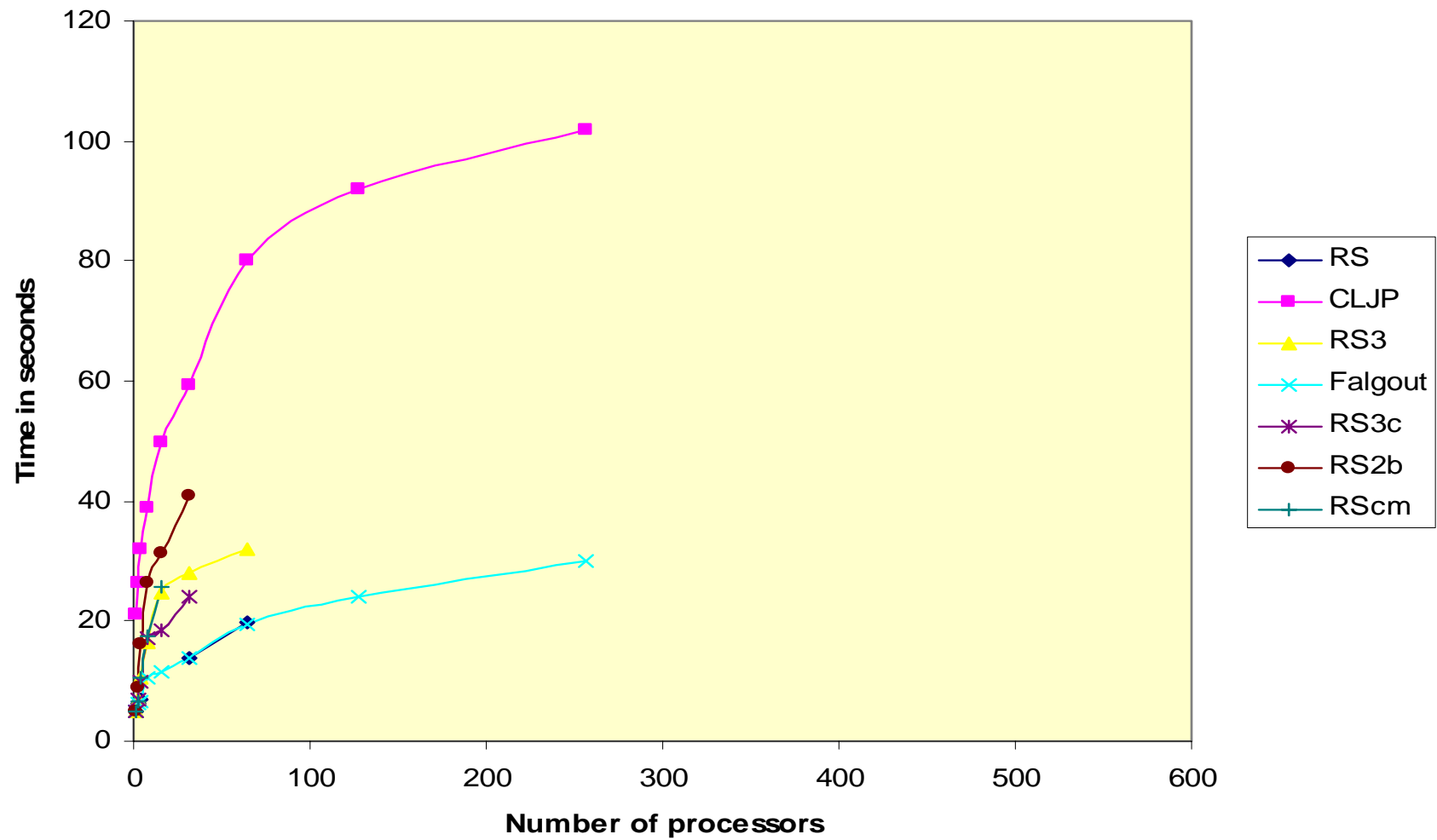
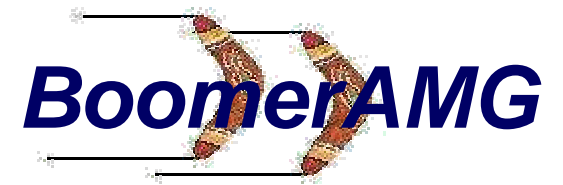
- **Gauß-Seidel**

$$(D-L)x^{(n+1)} = Ux^{(n)} + b \quad \text{where} \quad A = D - L - U$$

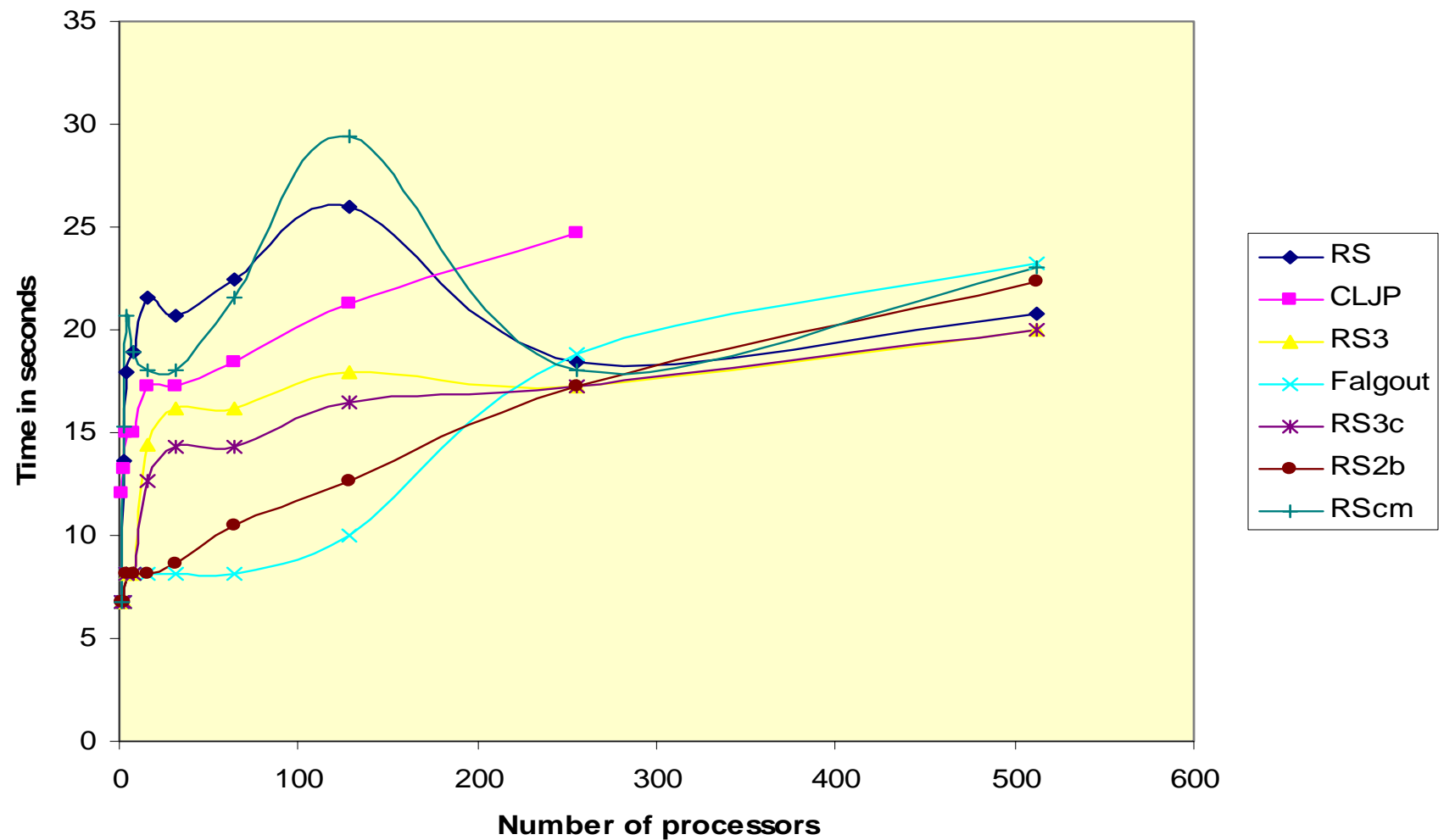
- **chaotic Gauß-Seidel**

use new values when available, old values on processor boundaries

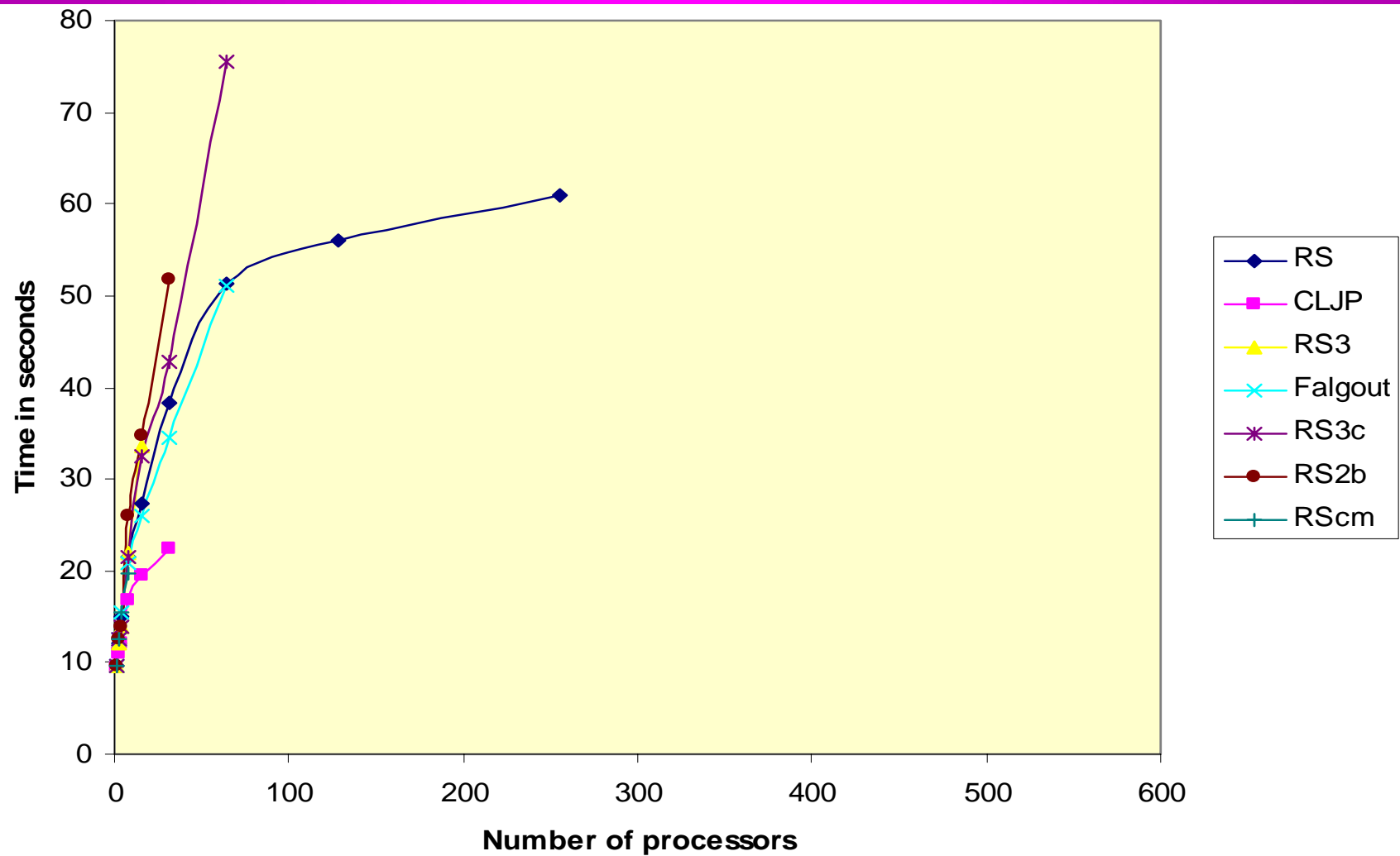
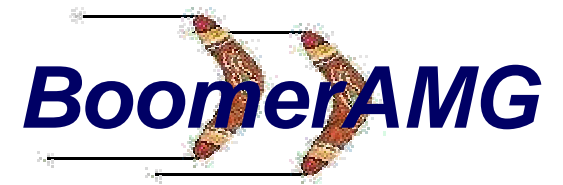
7 point 3D Laplacian: Solve times chaotic GS



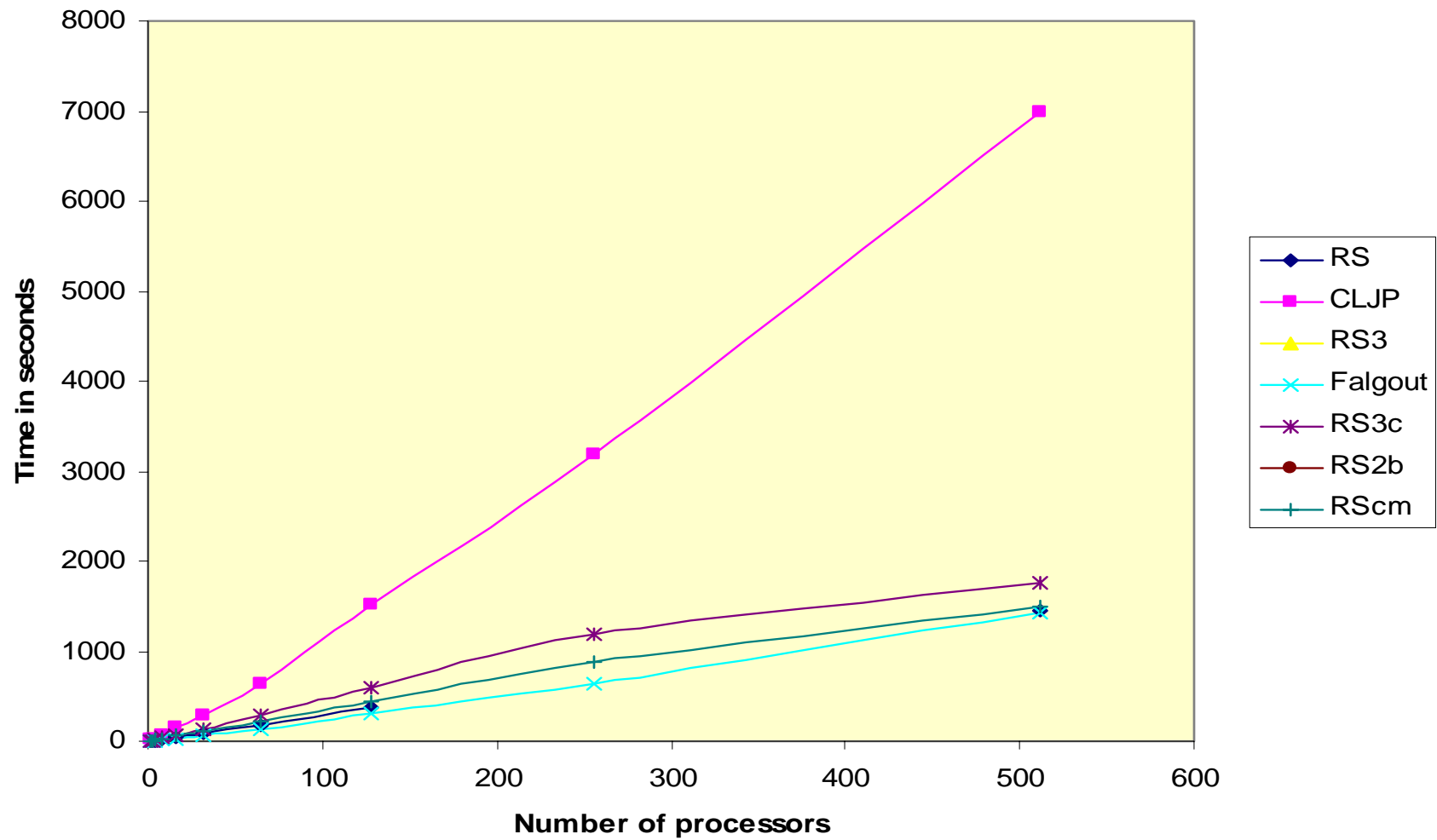
9 point 2D Laplacian: Solve times chaotic GS



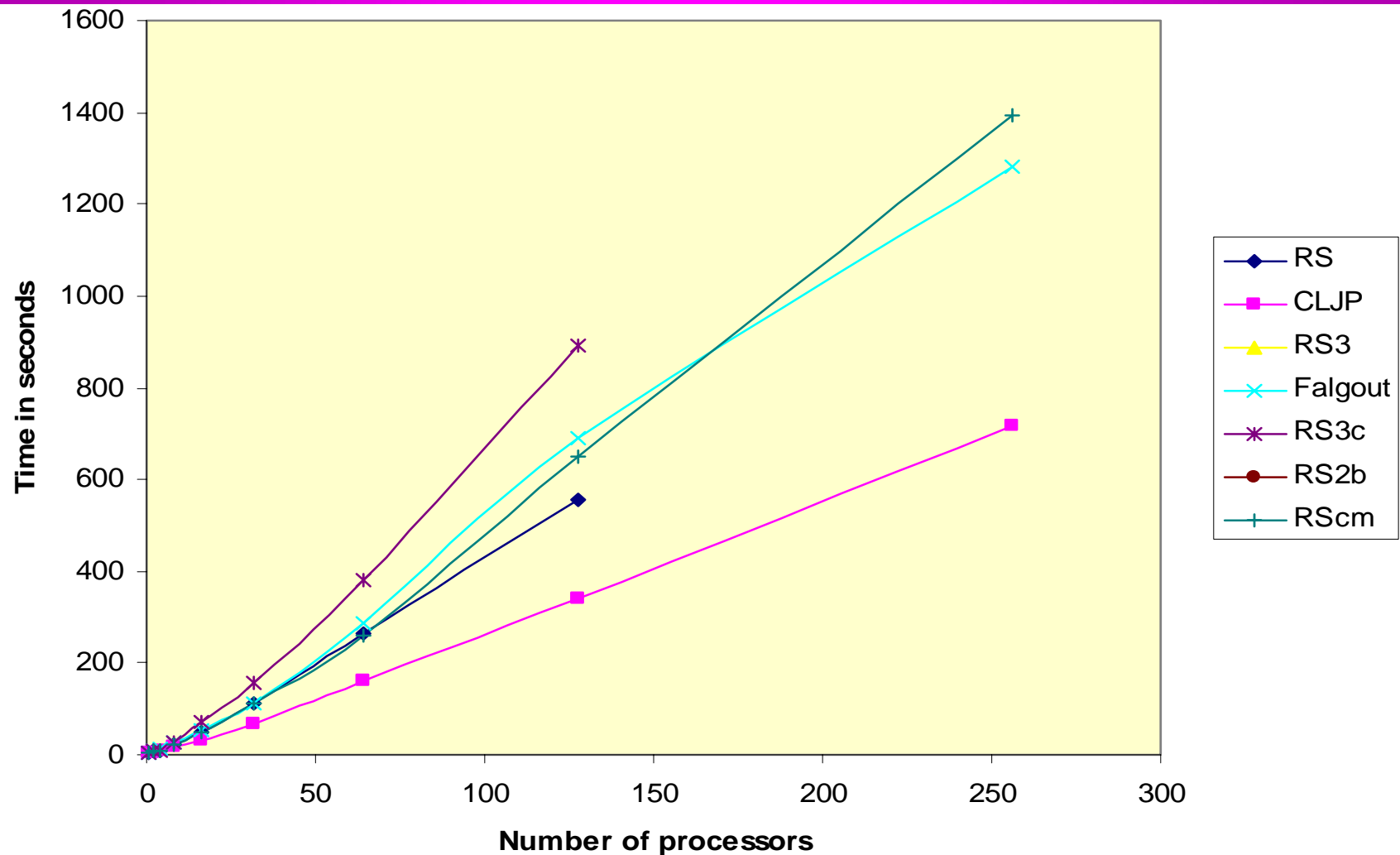
27 point 3D Laplacian: Solve times chaotic GS



7 point 3D Laplacian: Solve times Gauß-Seidel

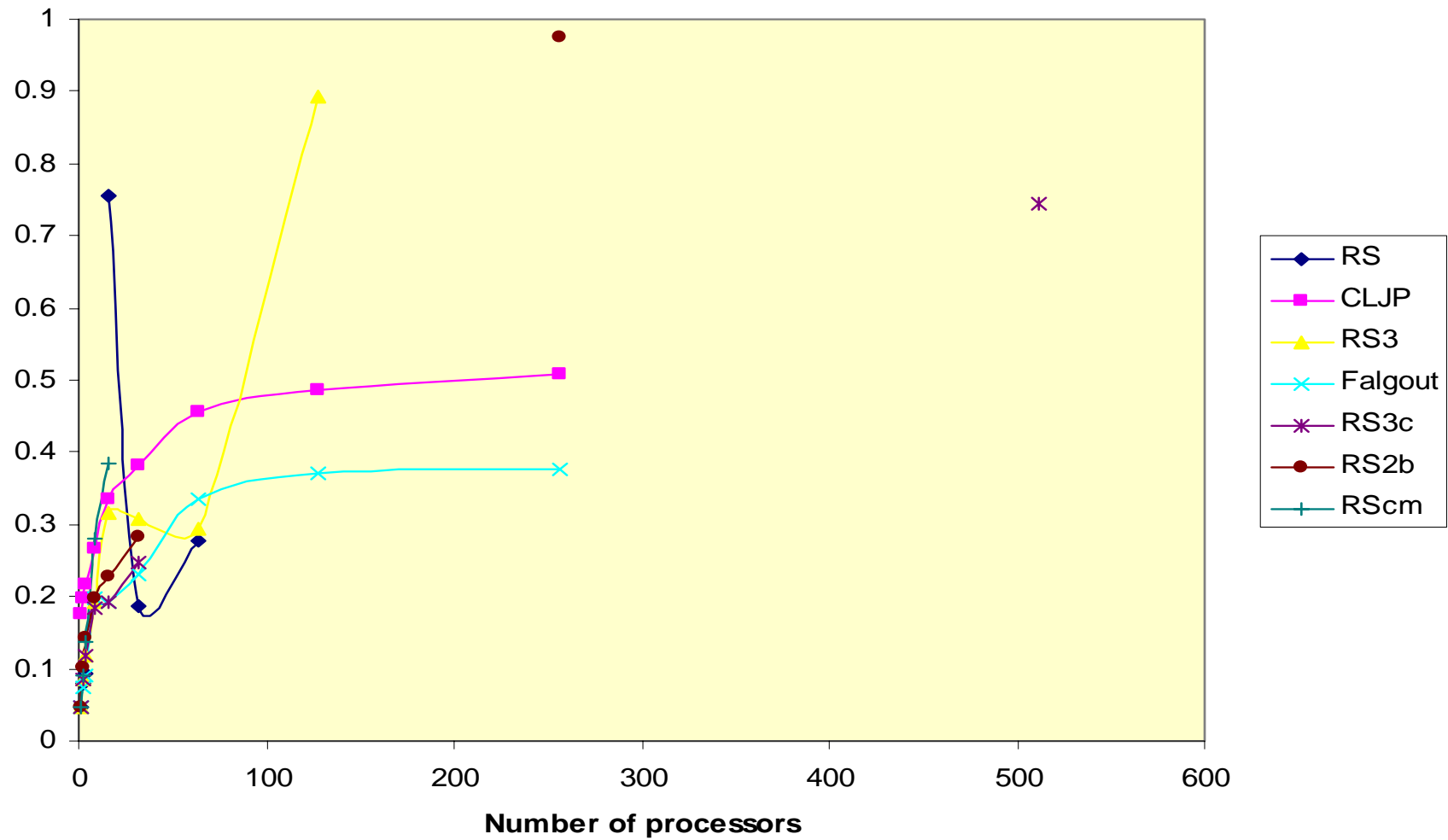


27 point 3D Laplacian: Solve times Gauß-Seidel

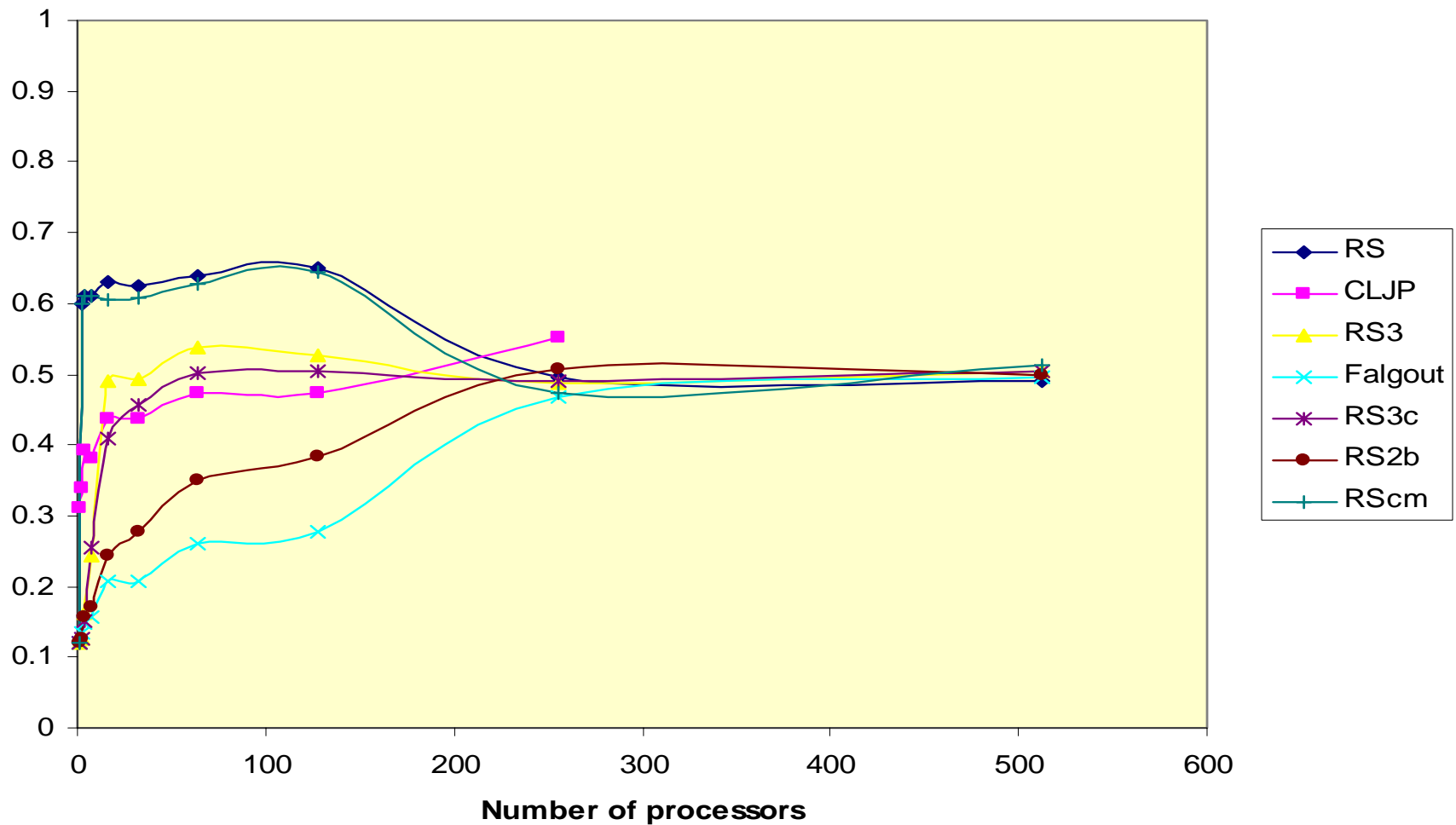
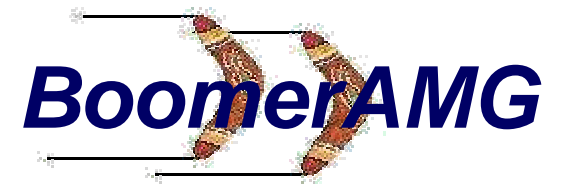


7 point 3D Laplacian: As. Conv. Factor chaotic GS

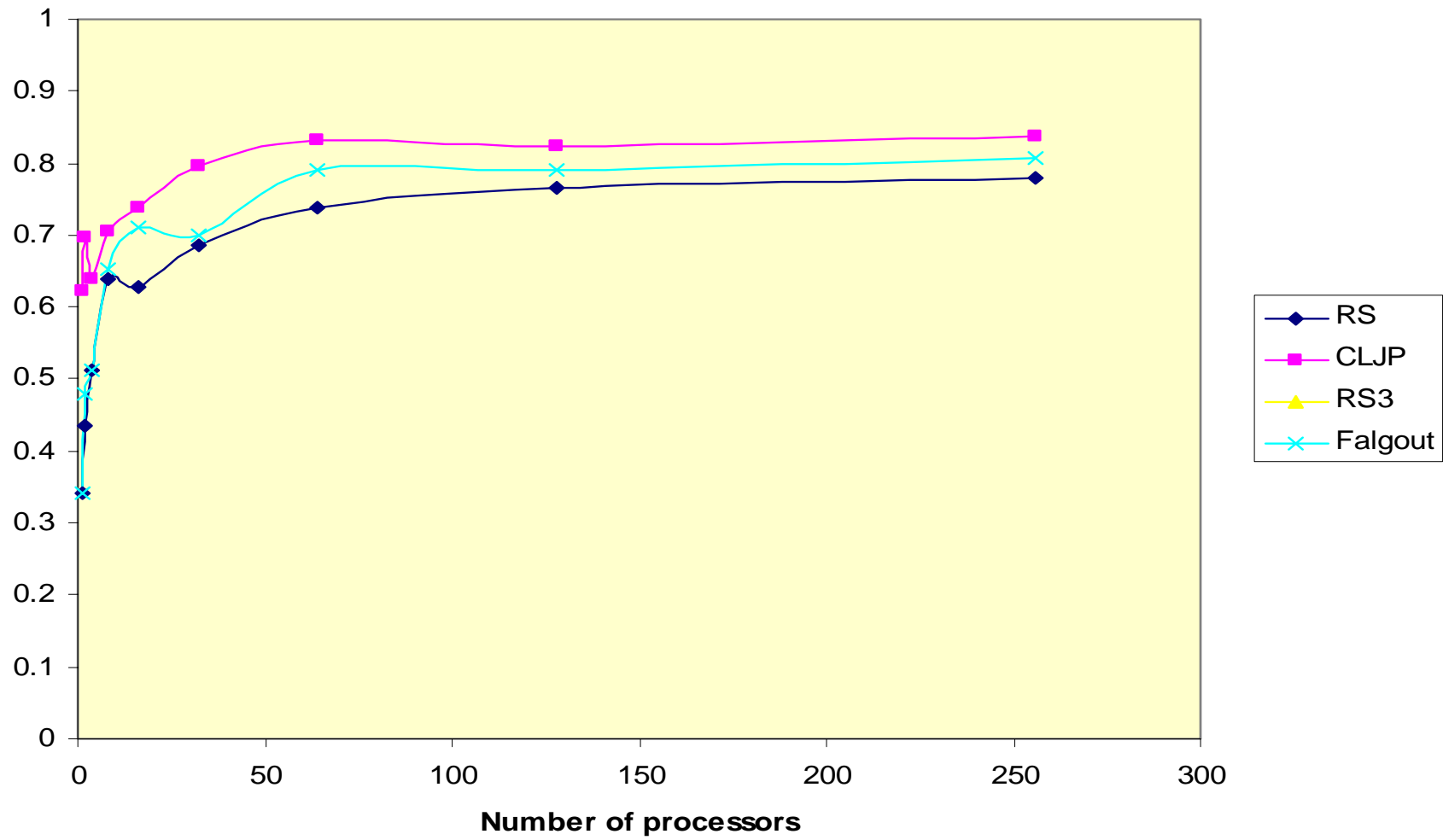
BoomerAMG



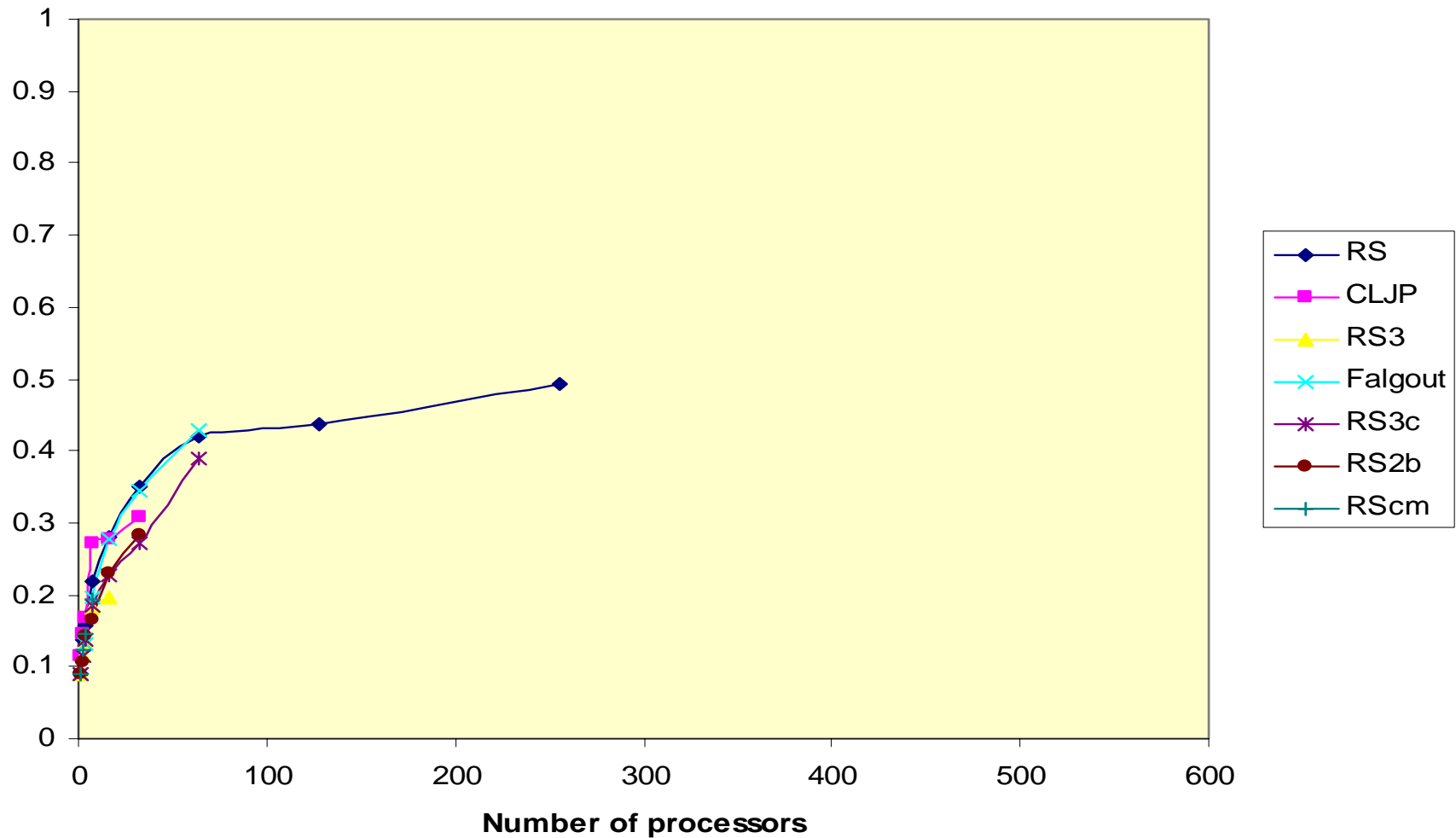
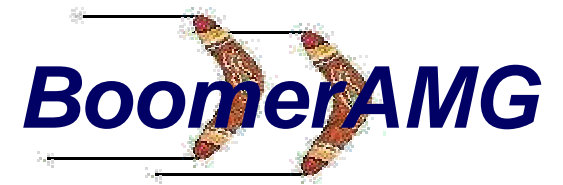
9 point 2D Laplacian: As. Conv. Factor chaotic GS



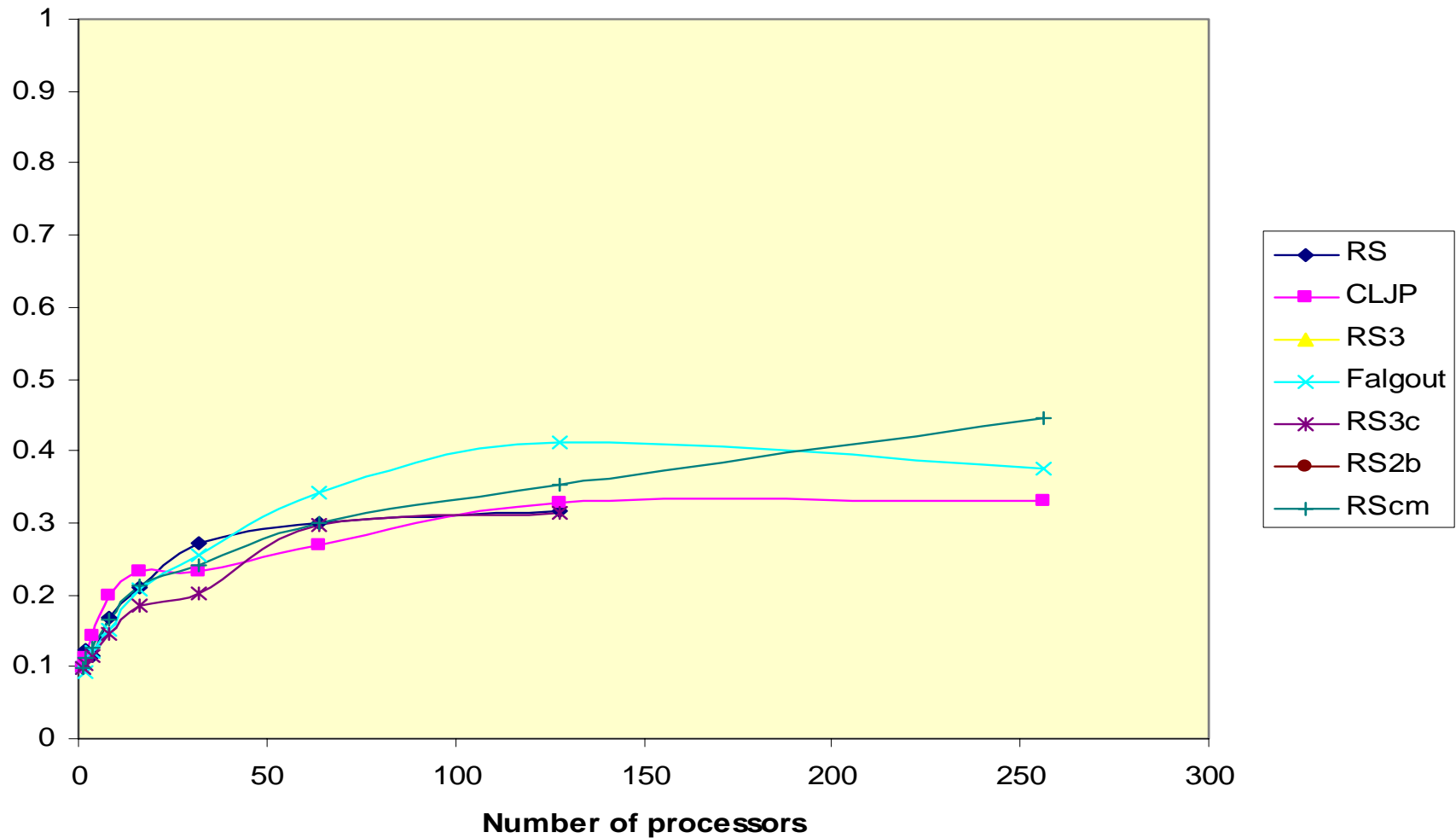
7 point 3D Laplacian: As. Conv. Factor wt. Jacobi



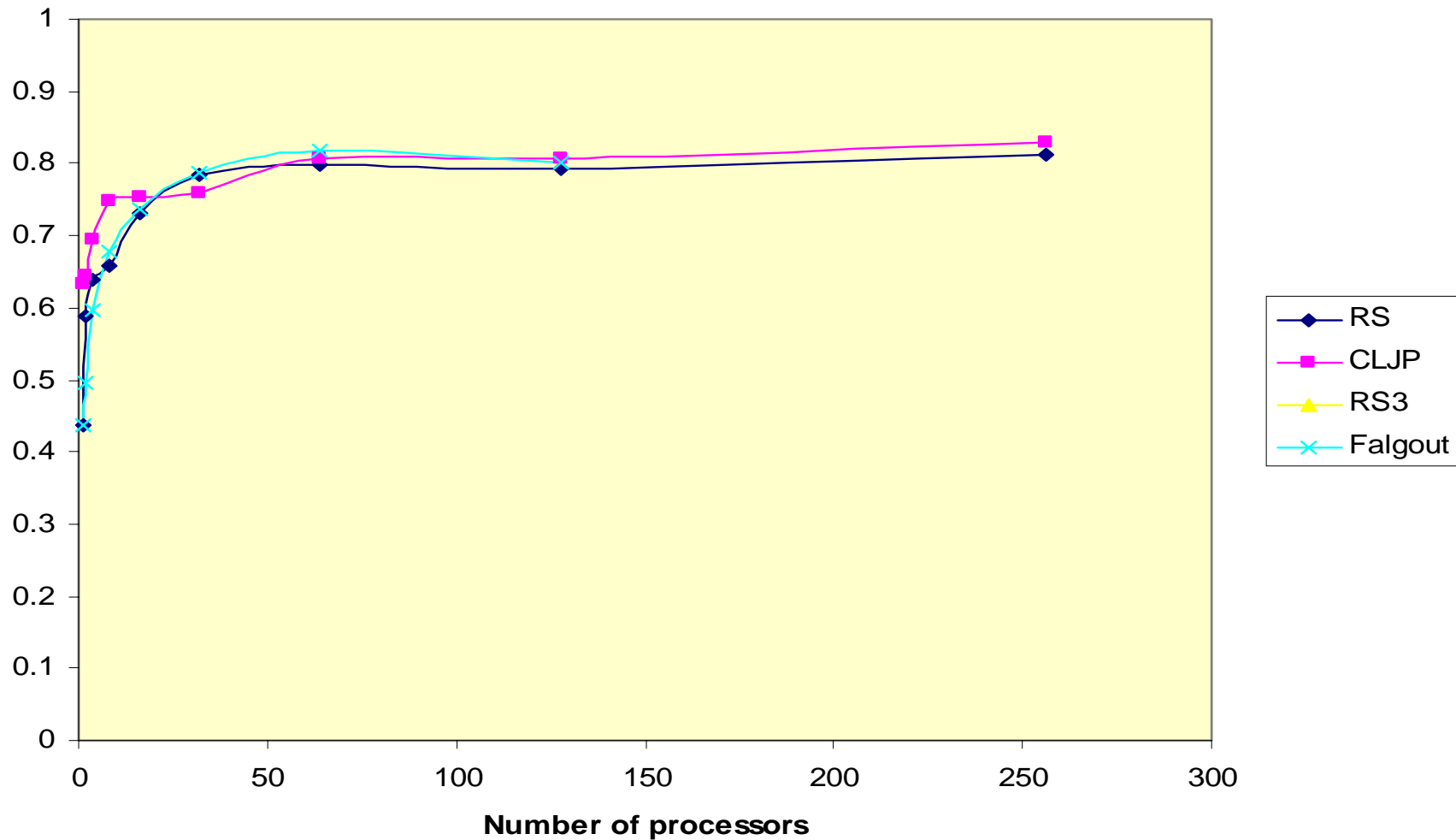
27 point 3D Laplacian: As. Conv. Factor chaotic GS



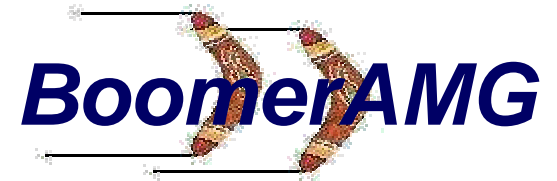
27 point 3D Laplacian: As. Conv. Factor GS



27 point 3D Laplacian: As. Conv. Factor wt. Jacobi



Conclusions & Future Work



- **AMG has been parallelized. It shows reasonably good scalability.**
- **Testing is still needed to implement the algorithms efficiently; to determine better ways of treating processor boundaries, operator complexities, and growing convergence factors.**
- **Future computer science plans include load balancing and efficient cache useage.**
- **Future algorithmic development centers on implementing “system” solvers and determining MG components using the finite-element stiffness matrices**
- This work was performed under the auspices of the U. S. Department of Energy by Lawrence Livermore National Laboratory under contract number: **W-7405-Eng-48.**